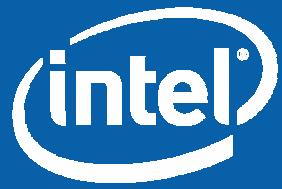


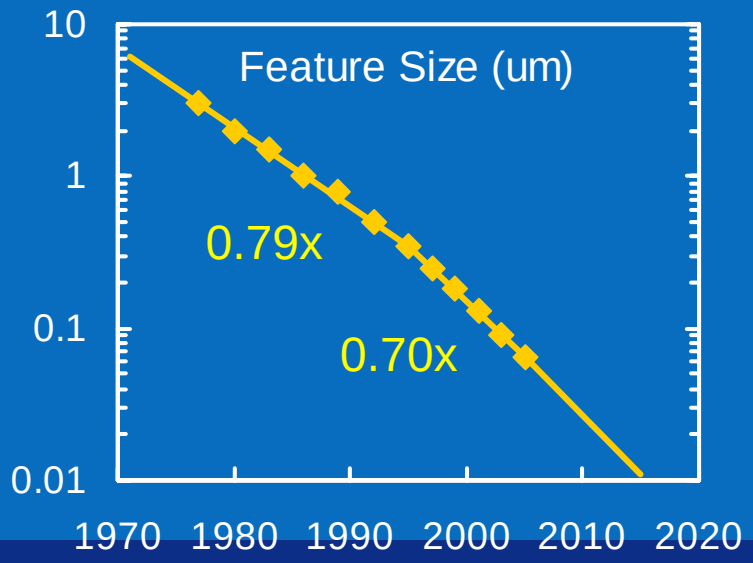
Why Intel is designing multi-core processors

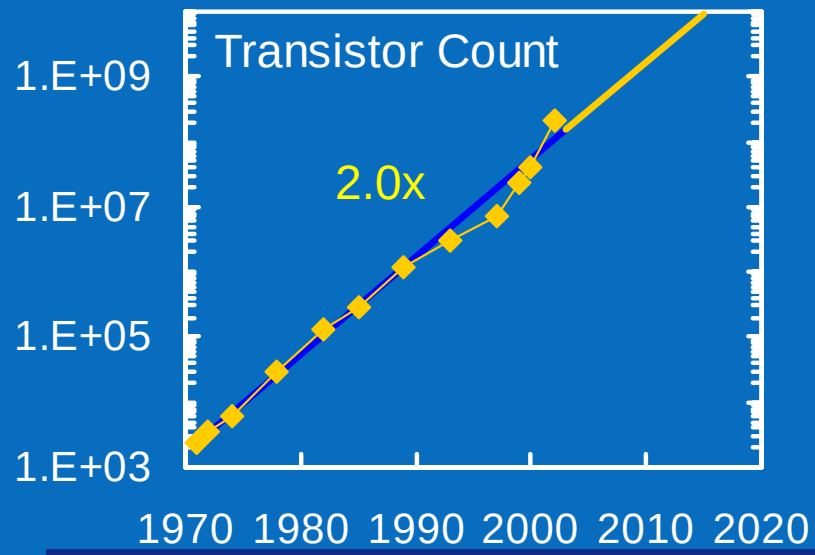
Geoff Lowney

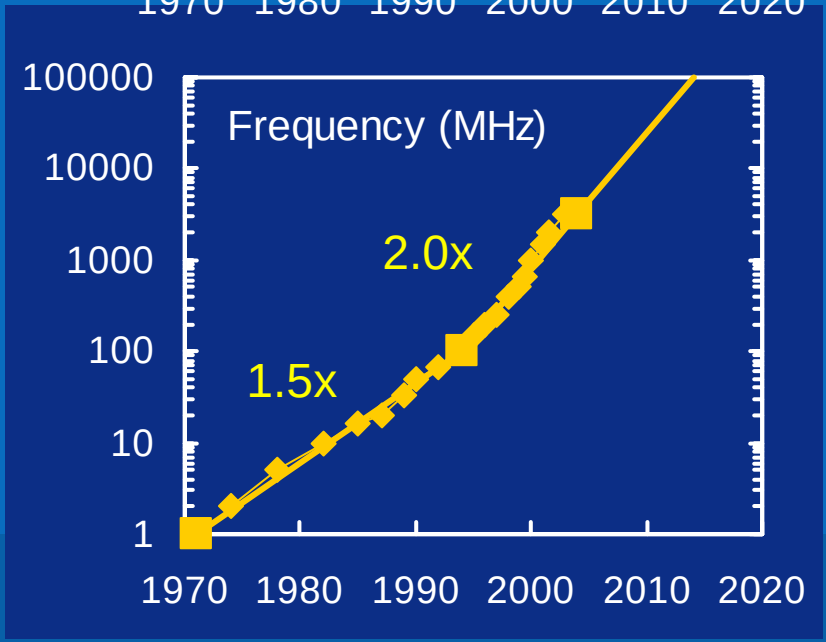
Intel Fellow, Microprocessor Architecture and Planning

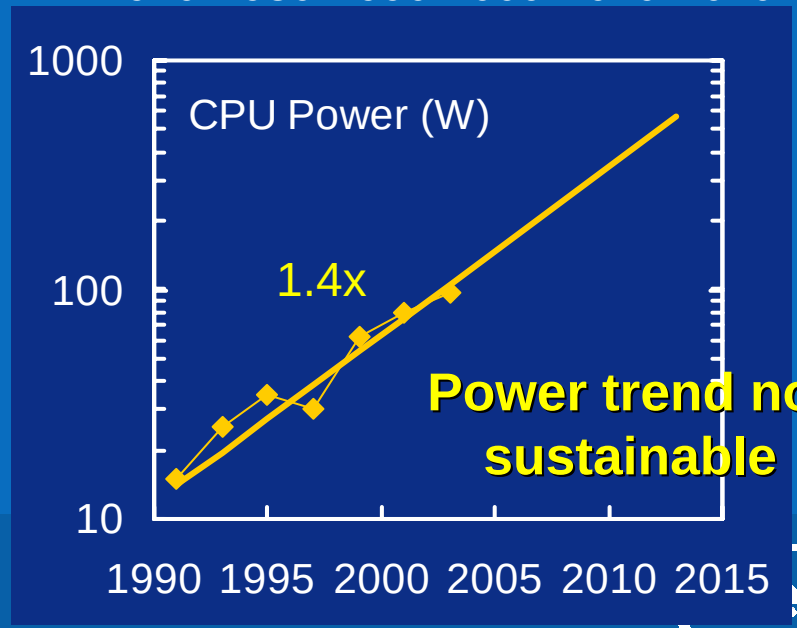
May, 2006

Moore's Law at Intel 1970-2005



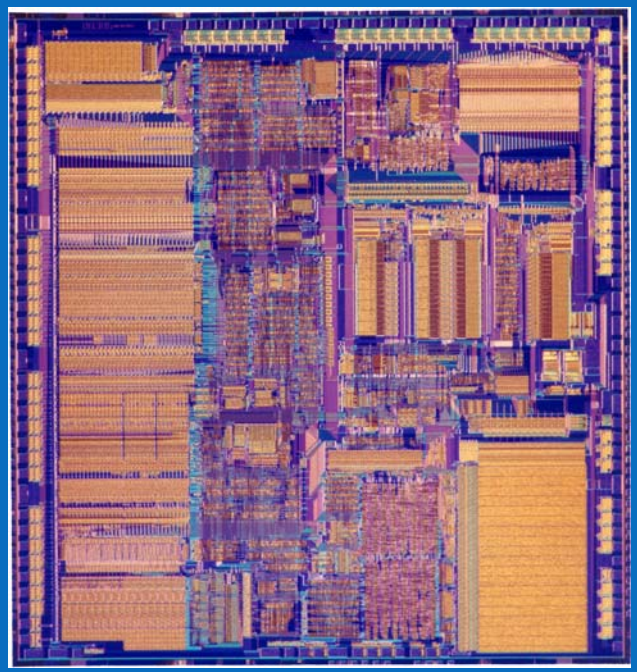


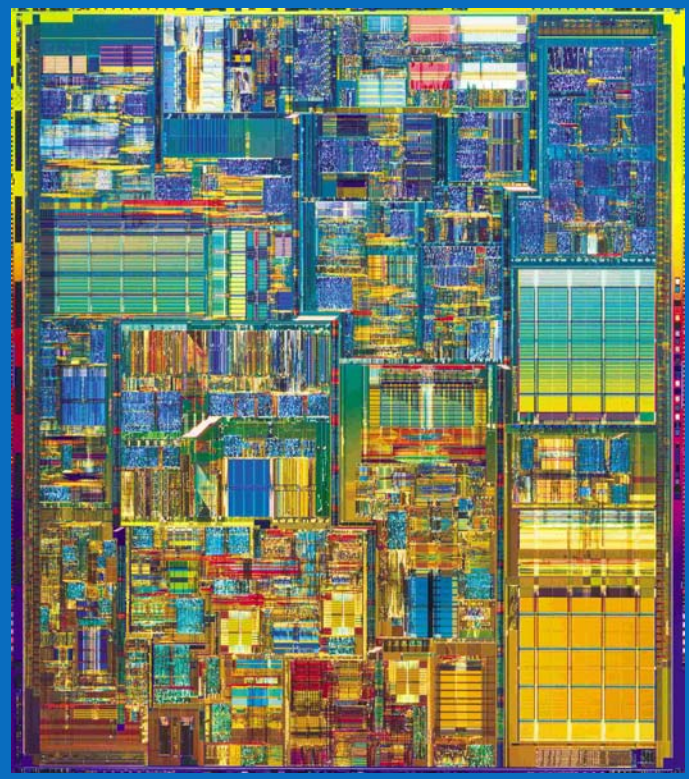




Pentium® 4 Processor

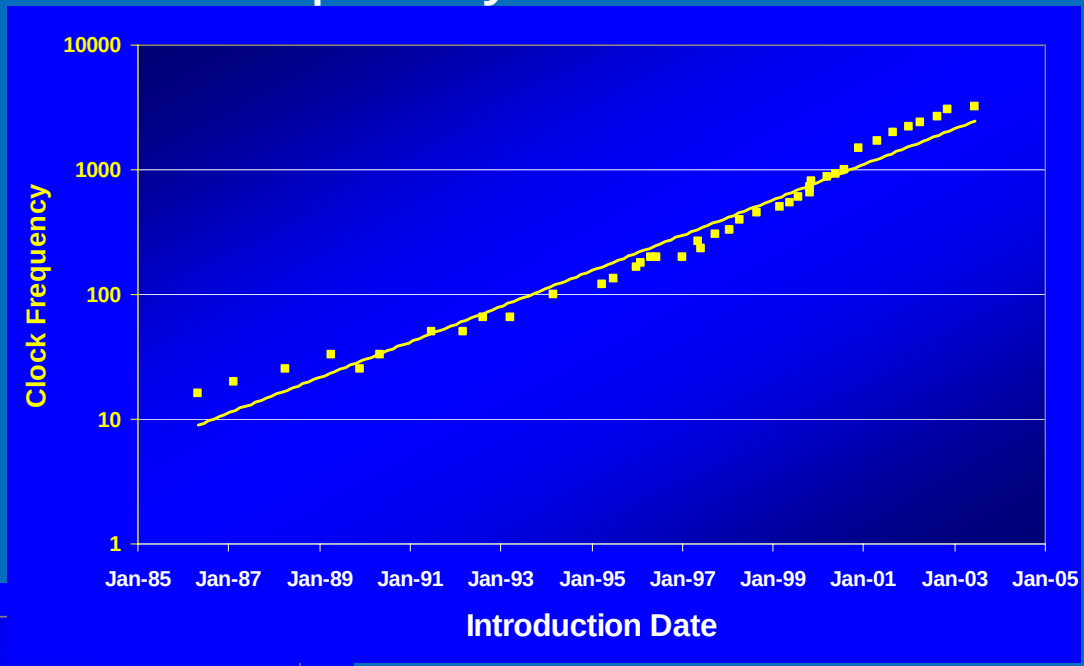
386 Processor



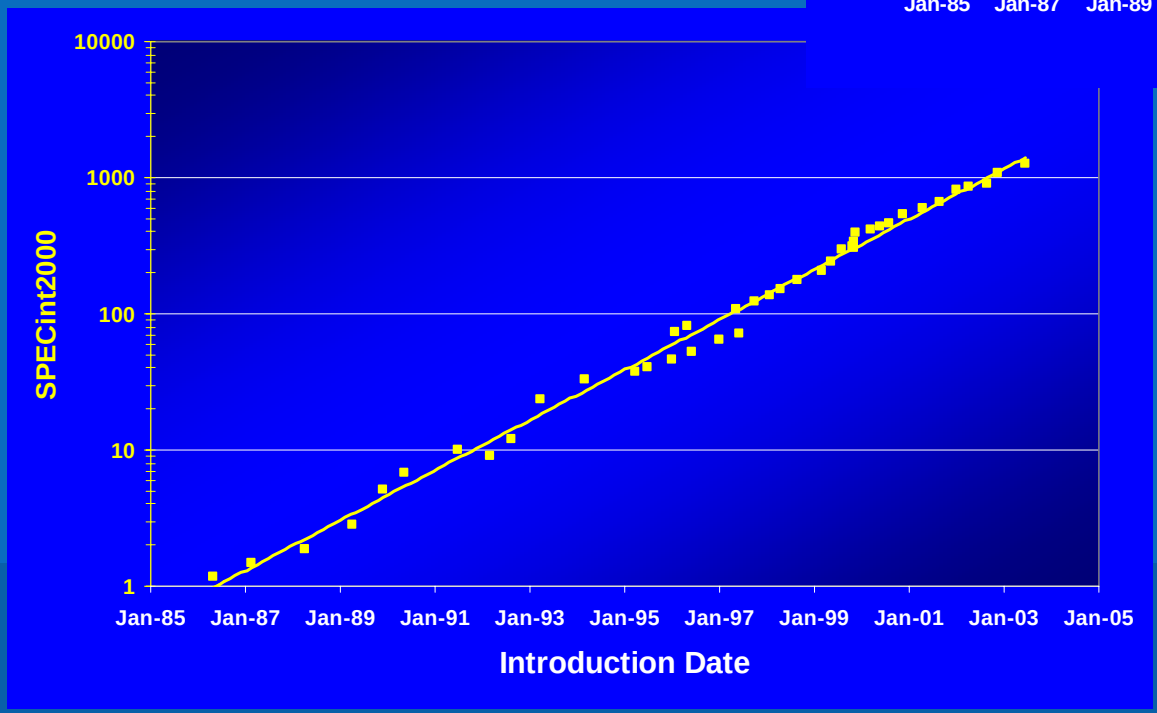


May 1986	17 Years	August 27, 2003
@16 MHz core	200x	@3.2 GHz core
275,000 1.5µ transistors	200x/11x	55 Million 0.13µ transistors
~1.2 SPECint2000	1000x	1249 SPECint2000

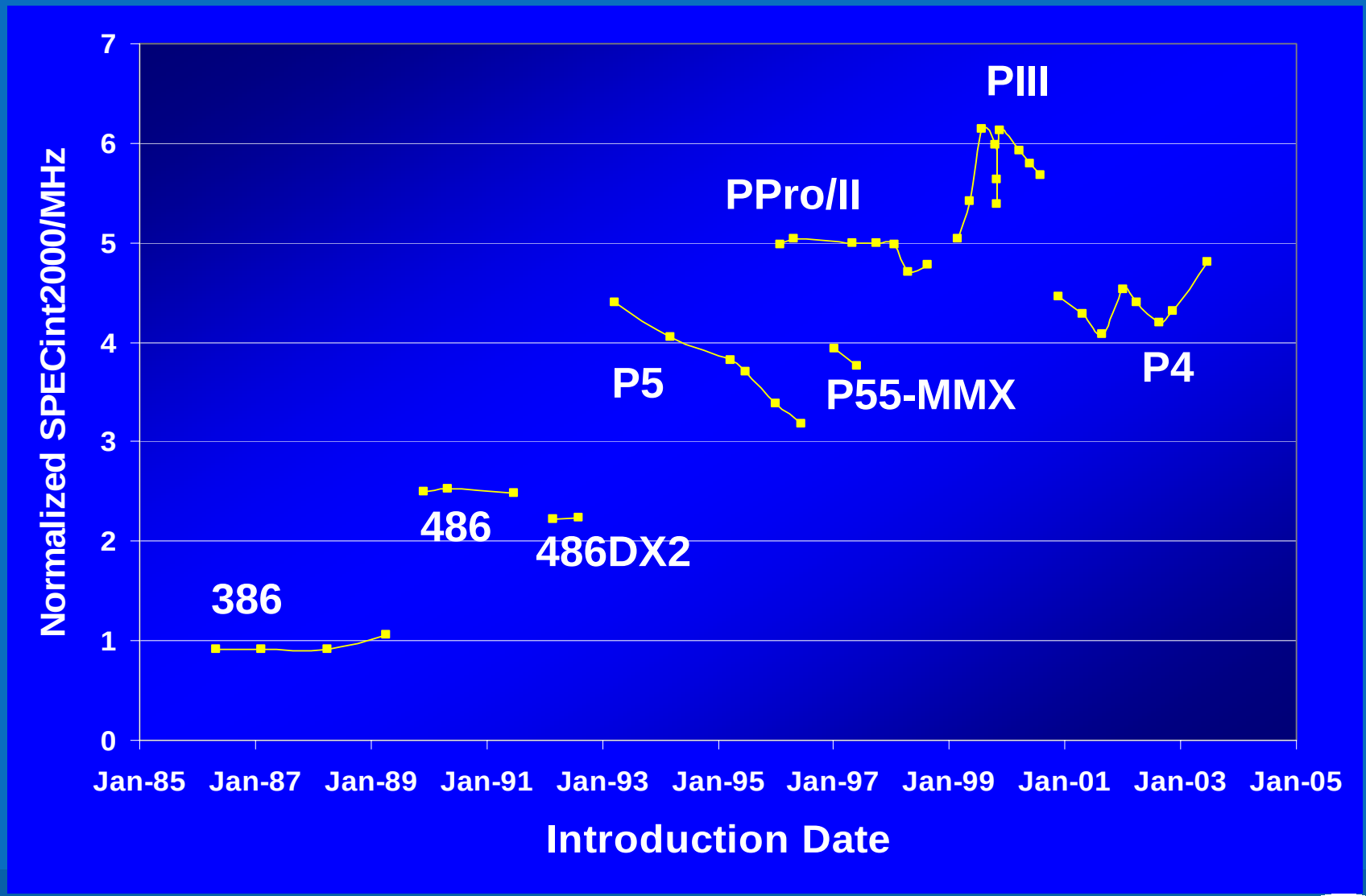
Frequency: 200x



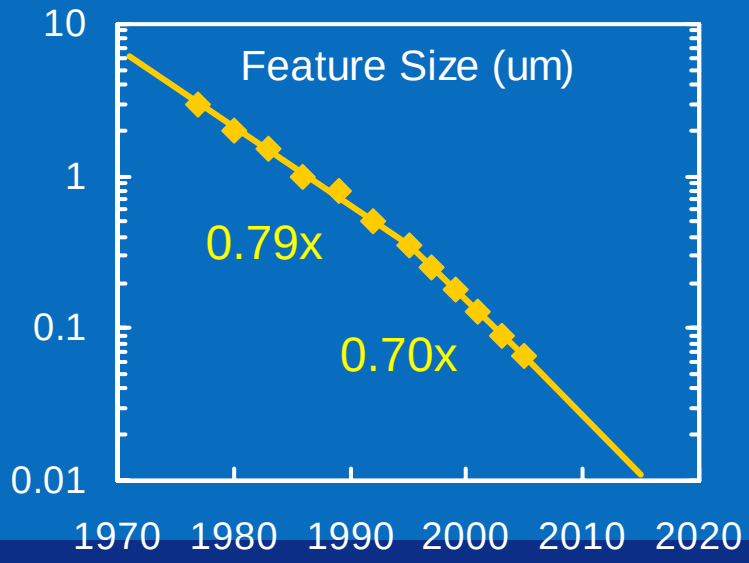
Performance: 1000x

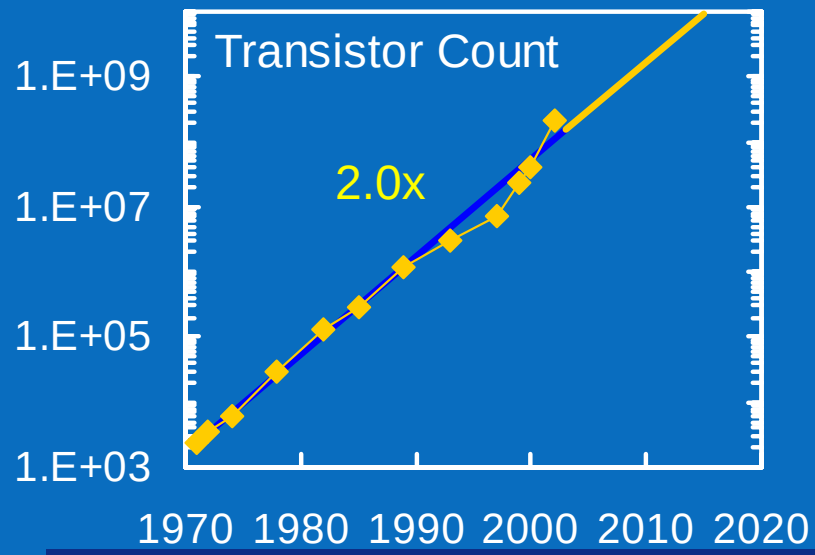


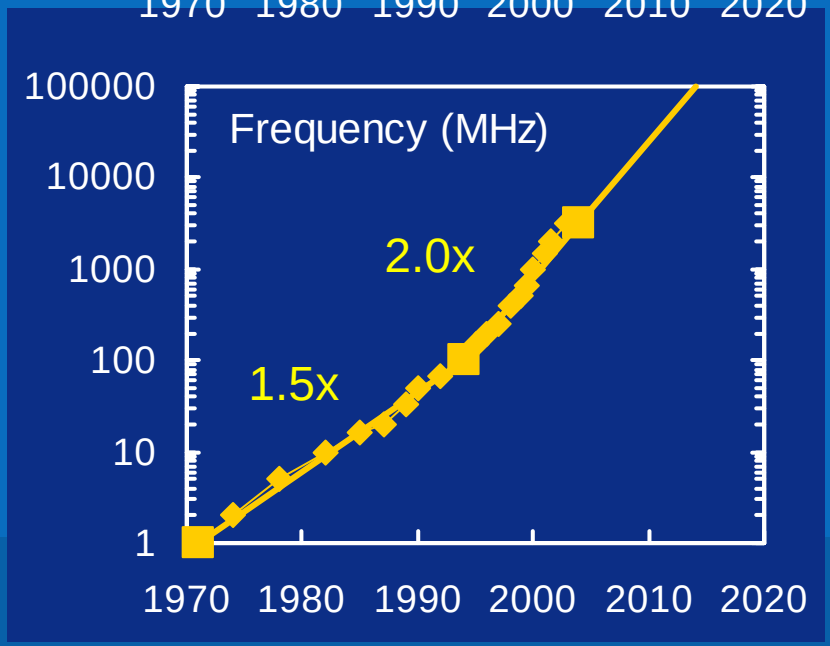
SPECint2000/MHz (normalized)

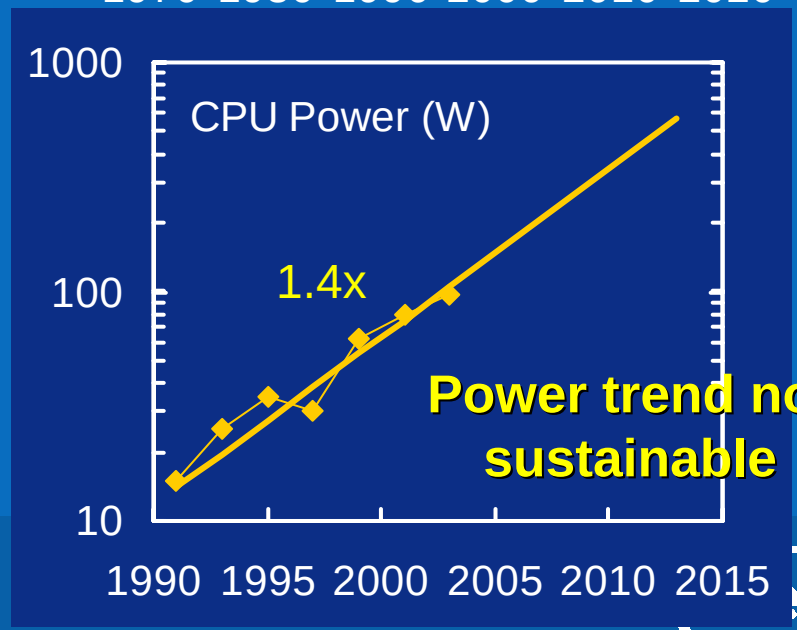


Moore's Law at Intel 1970-2005









Reducing power with voltage scaling

Power = Capacitance * Voltage**2 * Frequency

Frequency ~ Voltage in region of interest

Power ~ Voltage ** 3

10% reduction of voltage yields

- 10% reduction in frequency
- 30% reduction in power
- Less than 10% reduction in performance

Rule of Thumb

Voltage	Frequency	Power	Performance
1%	1%	3%	0.66%

Dual core with voltage scaling

RULE OF THUMB

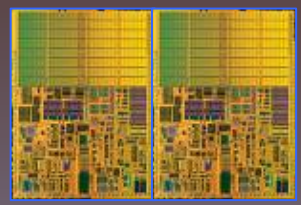
A 15% Reduction In Voltage Yields

Frequency Reduction	Power Reduction	Performance Reduction
15%	45%	10%

SINGLE CORE

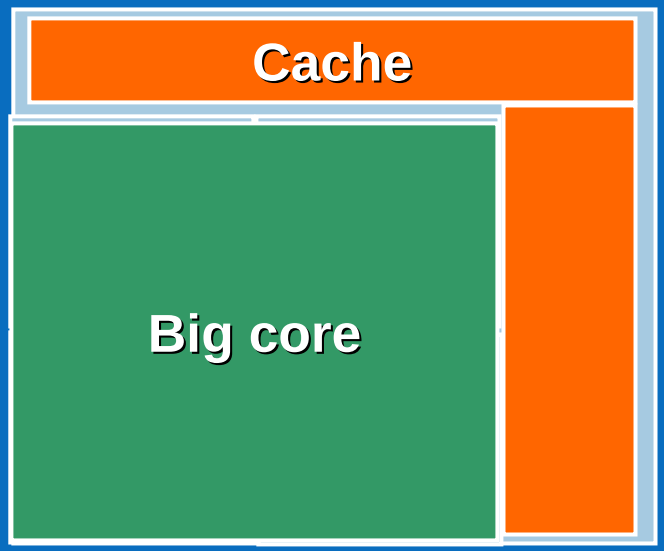
Area = 1
Voltage = 1
Freq = 1
Power = 1
Perf = 1

DUAL CORE



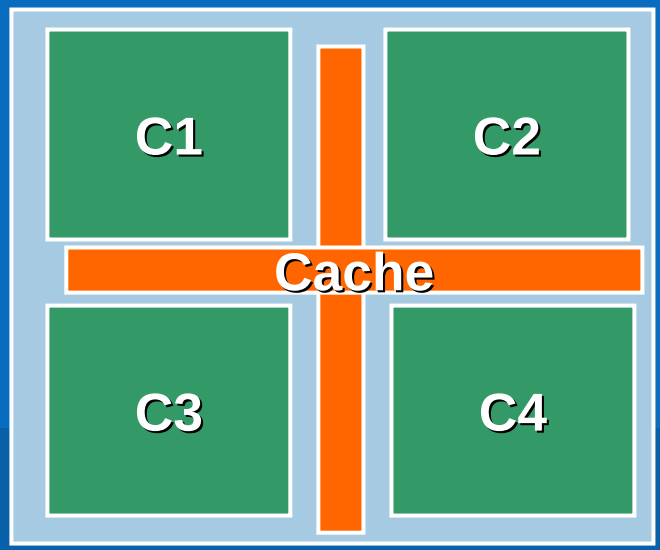
Area = 2
Voltage = 0.85
Freq = 0.85
Power = 1
Perf = ~1.8

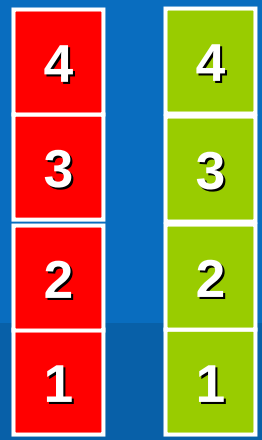
Multiple cores deliver more performance per watt



Power = ¼

Performance = 1/2





Many core is more power efficient

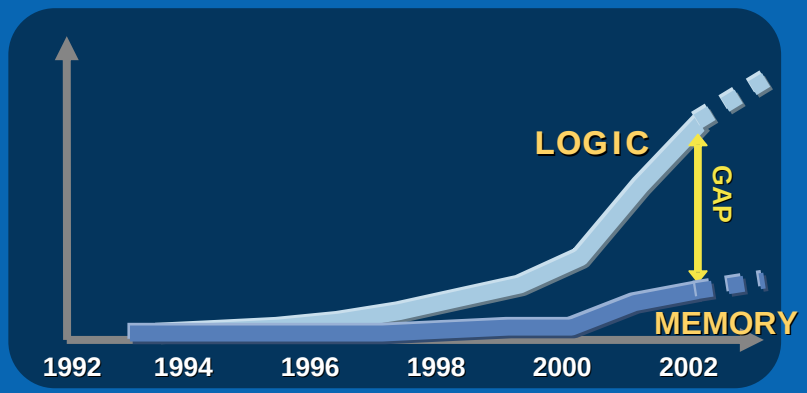
Power ~ area

Single thread performance ~ area.5**

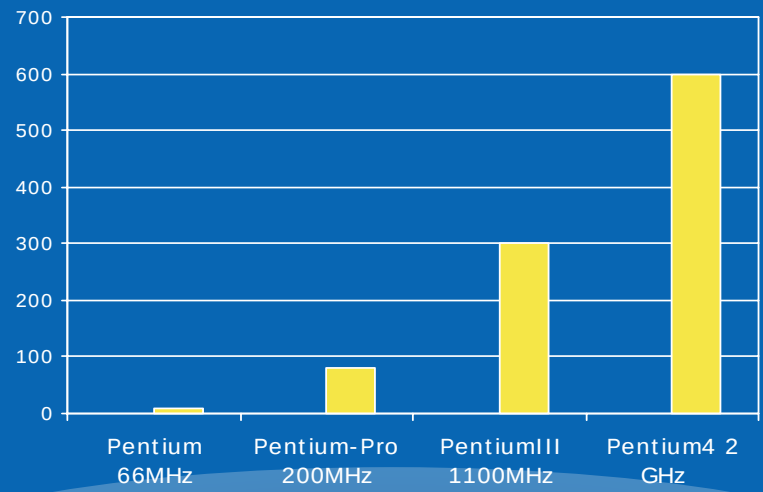
intel

Memory Gap

Growing Performance Gap



Peak Instructions Per DRAM Access



Reduce DRAM access with large caches

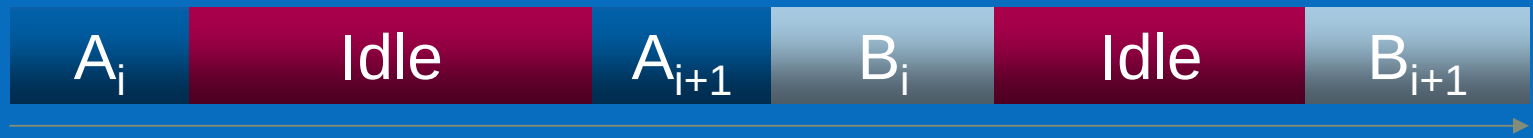
Extra benefit: power savings. Cache is lower power than logic

Tolerate memory latency with multiple threads

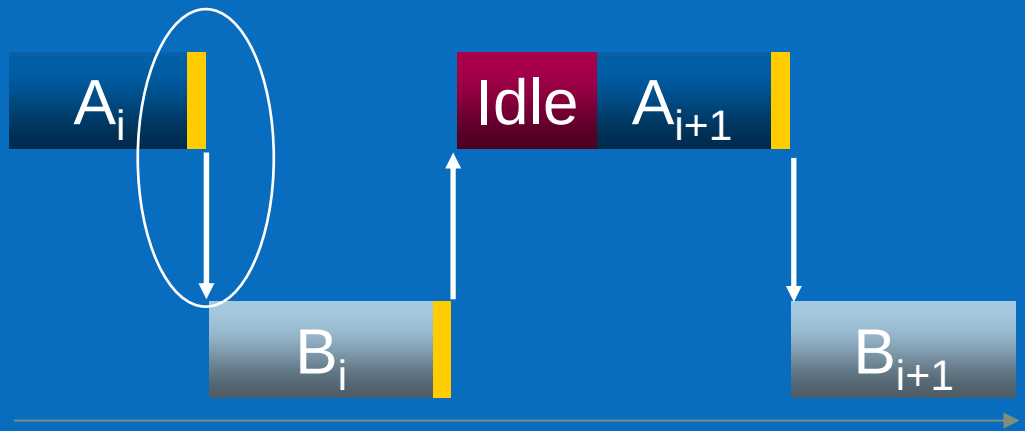
- Multiple cores
- Hyper-threading

Multi-threading tolerates memory latency

Serial Execution



Multi-threaded Execution

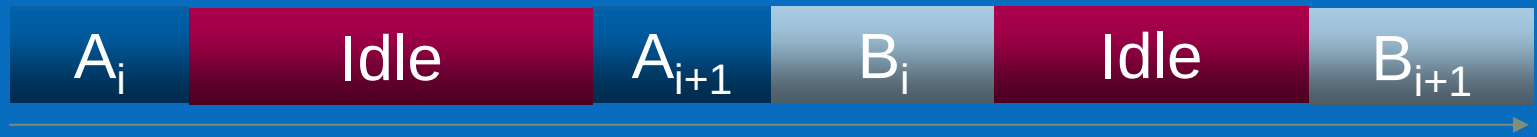


Execute thread B while thread A waits for memory

Multi-core has a similar effect

Multi-core tolerates memory latency

Serial Execution

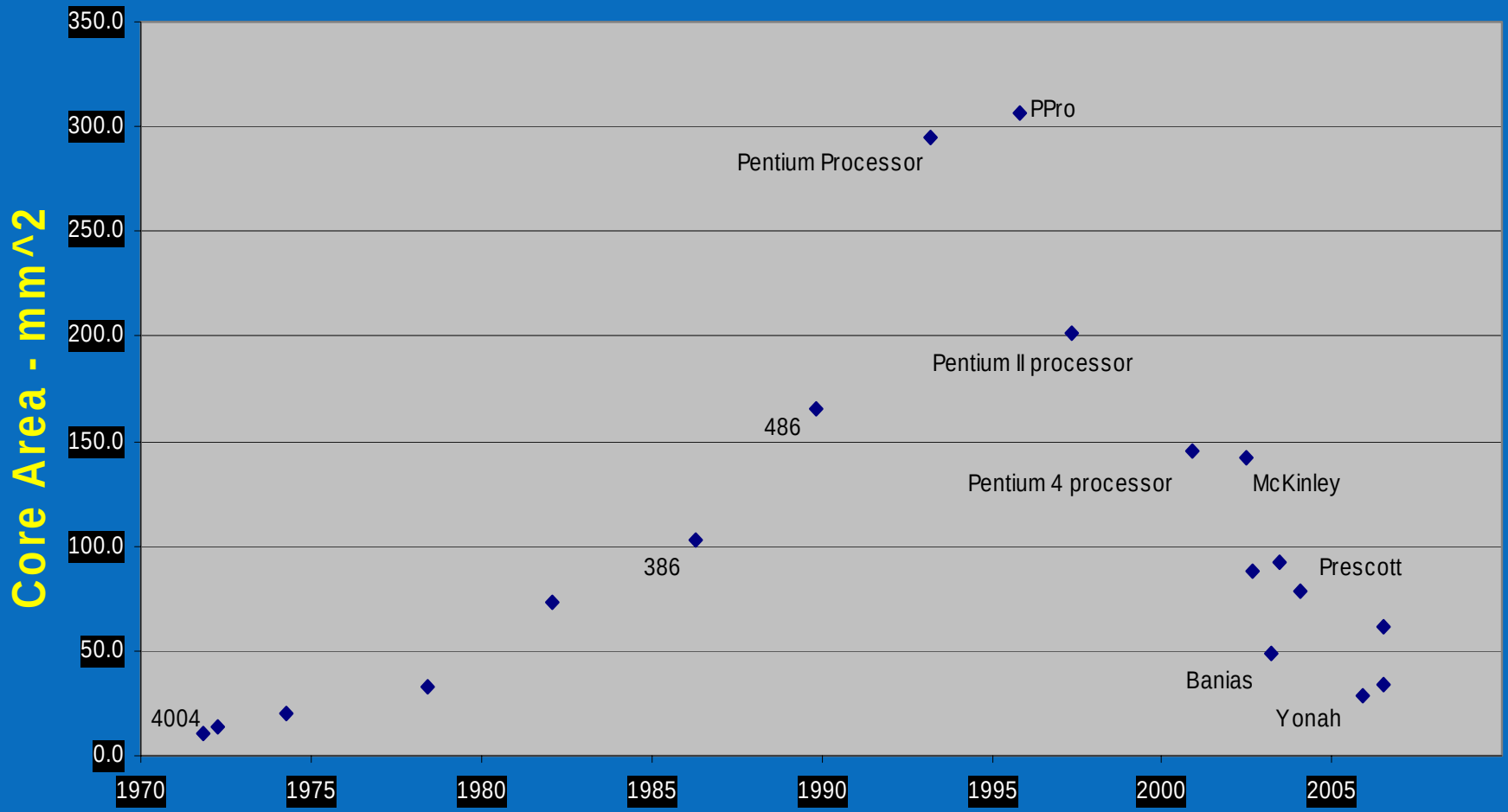


Multi-core Execution

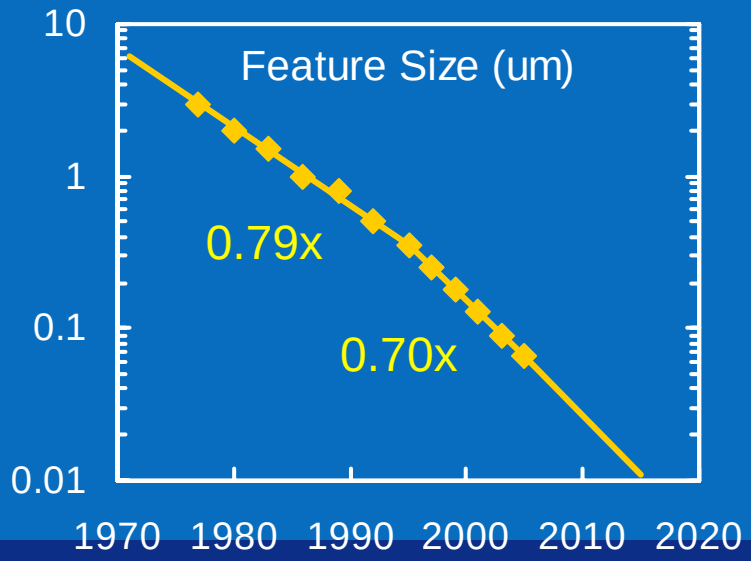
Execute thread A and B simultaneously

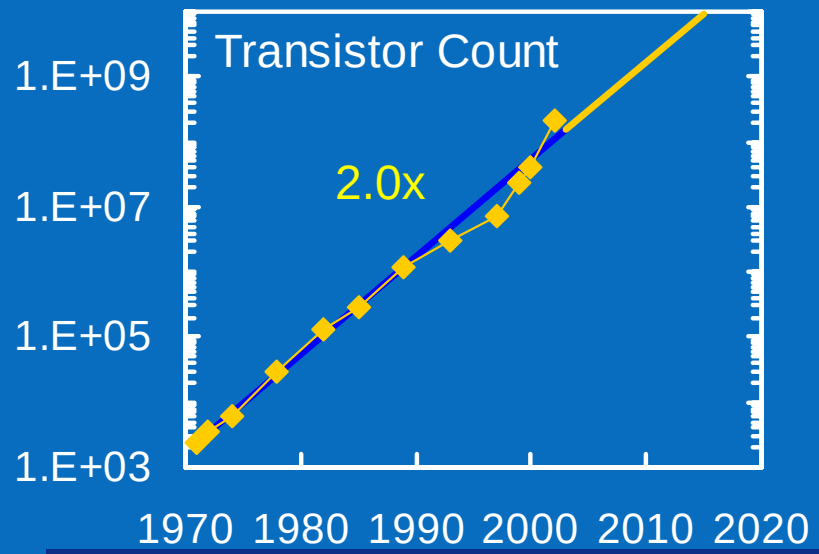
Core Area (with L1 caches) Trend

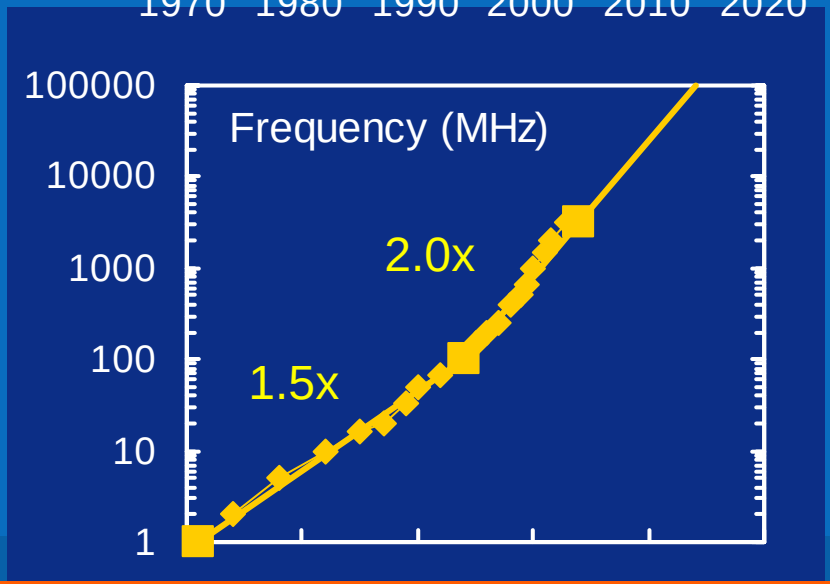
Die area/core shrinking after peaking with PPro

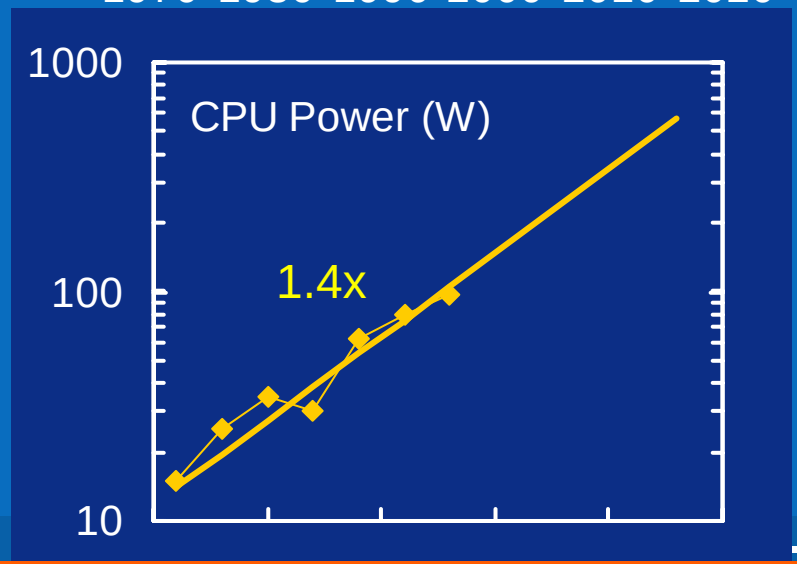


Moore's Law at Intel 1970-2005









CMP addresses power, memory, complexity

Moore's Law will provide transistors

Intel process technology capabilities

High Volume Manufacturing	2004	2006	2008	2010	2012	2014	2016	2018
Feature Size	90nm	65nm	45nm	32nm	22nm	16nm	11nm	8nm
Integration Capacity (Billions of Transistors)	2	4	8	16	32	64	128	256

Use transistors for

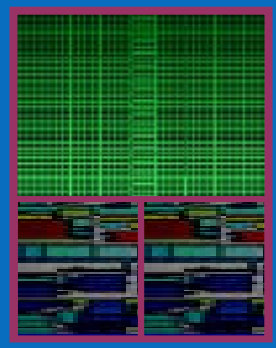
- **Multiple cores**
- **On-core memory (caches)**
- **New features (*Ts)**

Multiple cores and caches address power and memory latency issues

Next Gen Xeon® Microprocessor

- 2 highly efficient OOO cores
- 4M L2 cache
- 40% reduction in TDP power
- 80% performance improvement
- Server *Ts

Woodcrest



2H '06

Wider

- Higher FE bandwidth
- 4 wide decode
- 4 wide renaming
- 4 wide retire
- Additional Integer port
- 128 bit wide SSE2 implementation

Deeper Buffers

- Larger RS, ROB
- Larger SB
- More Line Fill buffers (MEM)

More Efficient Pipeline

- Macro-Fusion: CMP+JMP in 1 clock
- Enhanced Micro-op-fusion
- Cache to cache transfer in CMP
- FIFO scheduling in RS
- Pseudo single cyc Branch Predict
- Faster string instructions (REP mov)

Future Architecture
More Cores

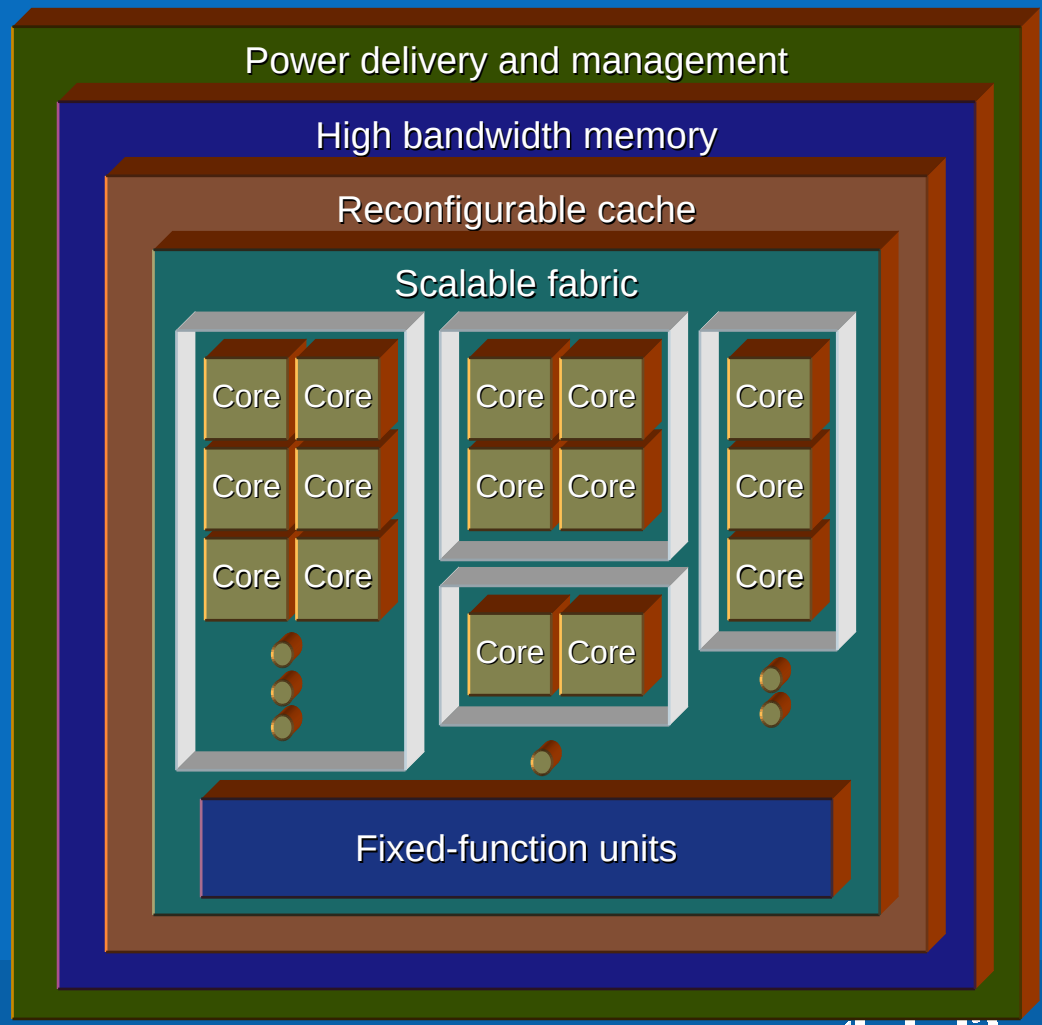
Open issues

Cores
- How many?
- What size?
- Homogeneous?

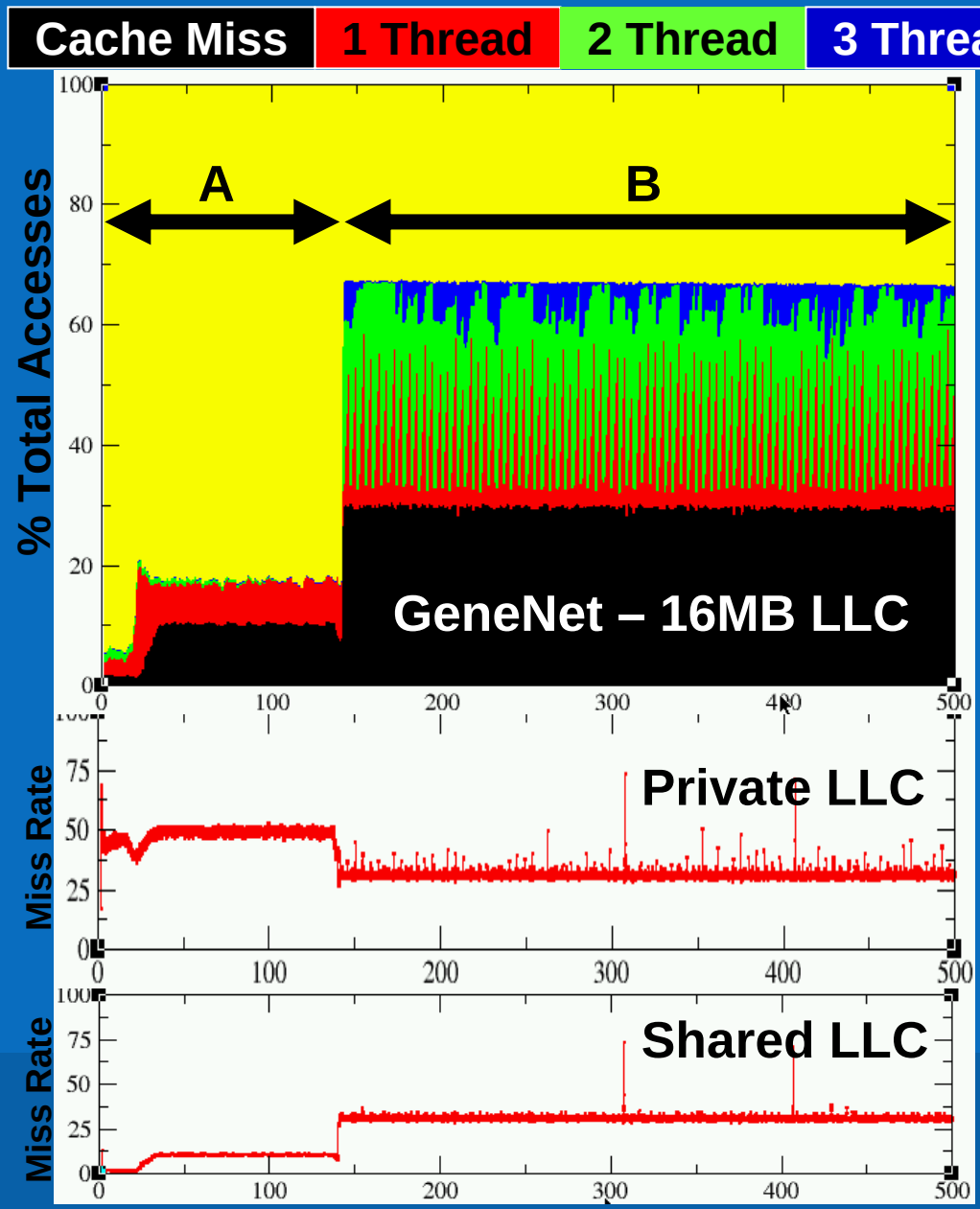
On-die Interconnect
- Topology
- Bandwidth

Cache hierarchy
- Number of levels
- Sharing
- Inclusion

Scalability



Shared Refs & Shared Caches…



Phase A: Shared caches perform better than private caches (25%)

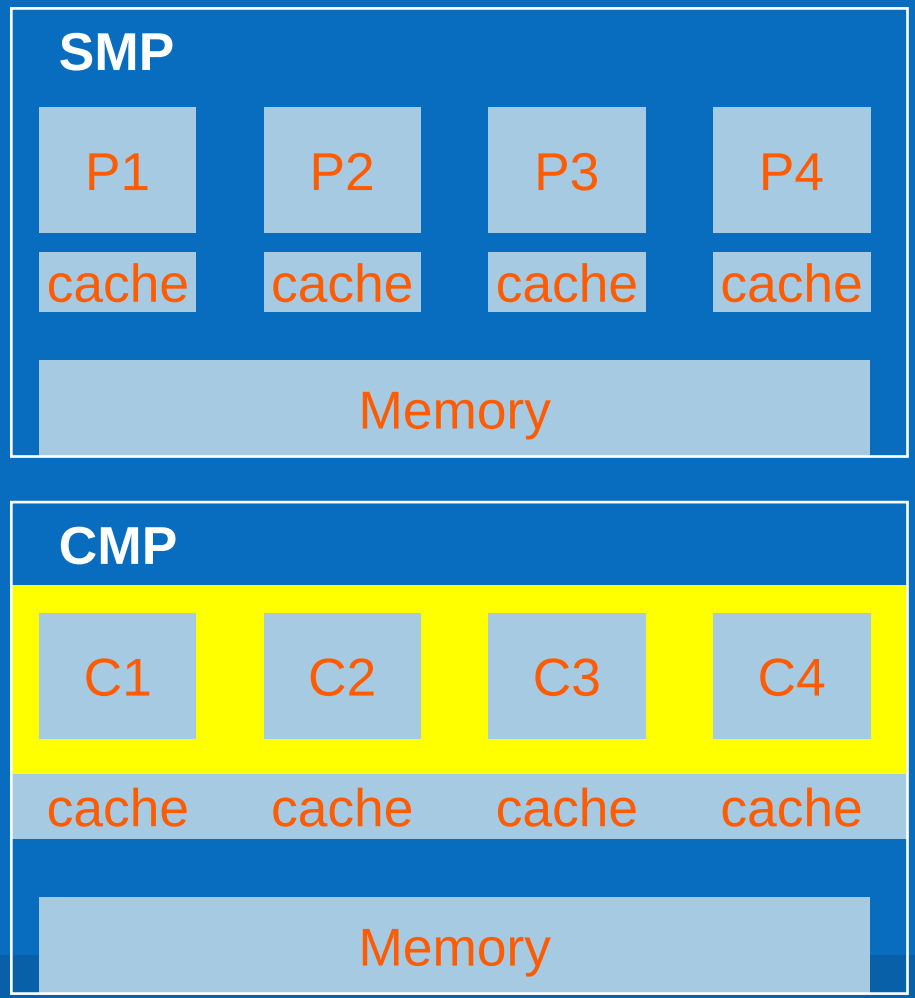
Phase B: Shared caches marginally better than private caches (5%)

Shared caches BETTER when shared data frequently referenced

Most workloads frequently reference shared data

From: [Jaleel, et al. 2006]

How does Multicore Change Parallel Programming?



No change in fundamental programming model

Synchronization and communication costs greatly reduced

- Makes it practical to parallelize more programs

Resources now shared

- Caches
- Memory interface
- Optimization choices may be different

Summary

Technology is driving Intel to build multi-core processors.

- Power
- Memory latency
- Complexity

Parallel programming is a central issue.

Parallel applications will become mainstream